SPM project: Parallel Prefix

Gaspare Ferraro, 520549

Master Degree in Computer Science - University of Pisa

ferraro@gaspa.re

November 1, 2018

1 Introduction

In this report we will analyze, theoretically and practically, the resolution of the problem of the (parallel) prefix sum:

Given a vector $x = \langle x_0, x_1, \ldots, x_{n-1} \rangle$ *and a binary operation* $\oplus$ *compute the vector* $y = \langle x_0, x_0 \oplus x_1, x_0 \oplus x_1 \oplus x_2, \ldots, x_0 \oplus x_1 \oplus \ldots \oplus x_{n-1} \rangle$.

In literature this operation is also called (inclusive) scan or partial sum.
For the analysis of the problem we have to make two assumptions:

- The binary operation $\oplus$ is associative ($a \oplus (b \oplus c) = (a \oplus b) \oplus c$) and commutative ($a \oplus b = b \oplus a$), this is an important assumption as we will see in the next chapters the order of the operations may not be preserved.
- The size of the input vector is a power of 2, not a strong assumption, as all the algorithms we will present could be easily generalize to all the sizes, but it only helps to simplify some operations.

2 Sequential algorithm

The sequential algorithm simple compute each element of the vector y as follow:

$$y_i = \begin{cases} x_0 & \text{for } i = 0 \\ x_i \oplus y_{i-1}, & \text{for } 0 < i < n \end{cases}$$

This algorithm is optimal in a sequential model as it has a running time of $\mathcal{O}(n)$, assuming that $\oplus$ is $\mathcal{O}(1)$, and performs $n-1$ calls to $\oplus$ operation.

3 Parallel architecture design

The problem of computing the prefix sum vector is a classical example of a problem that have an optimal solution in a sequential model but that can be optimized in a parallel model. The optimization is not in terms of total complexity or in the number of $\oplus$ operations performed (which are already optimals in the sequential algorithm) but in terms of completion time. When the number of available threads is more than one we can trade-off more total work for less completion time.

We will now introduce two different algorithms that solve in an efficient way the prefix sum problem in a parallel model.

3.1 Block-based algorithm

The idea behind the first parallel algorithm is to compute the prefix vector in three phases:

- Partitionate the input vector in blocks and compute in parallel the prefix vector of each of them.
- In the second phase we put the final element of each block in a temporary vector and compute its prefix vector.
- Finally for each block i (except for the first one) add in parallel the $(i-1)$th element of the temporary vector.

At the end of the three phases the final vector contains the .

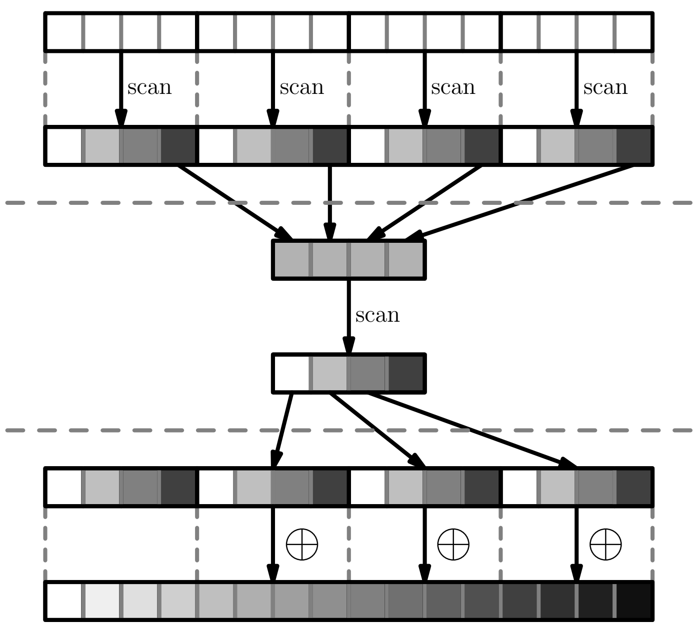


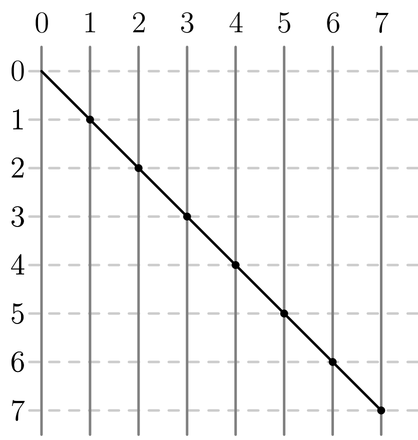
Figure 1: Block-based algorithm graphic representation

3.2 Circuit-based algorithm

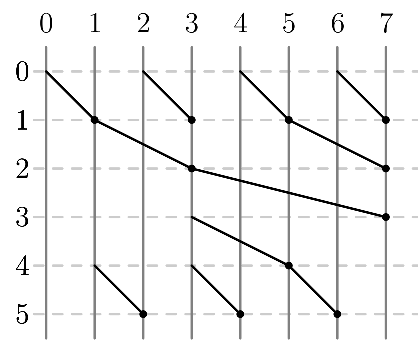
Another kind of algorithms are the one based on a circuit-like representation, there is plenty in literature of different class of prefix circuit.

A prefix circuit is a series of collection of operations

For example the serial algorithm seen before can be represented as a prefix circuit $S(n) = \{G_0, G_1, \ldots, G_{n-1}\}$ where $G_i = \{(i, i+1)\}$.



(a) Prefix circuit $S(8)$



(b) Prefix circuit $P(8)$

Figure 2: Examples of prefix circuits

4 Performance modeling

4.1 Block-based algorithm

4.2 Circuit-based algorithm

5 Implementations structure and details

All the implementations are written in $C++17$, the source code is available as attachment with the report or on GitHub

(https://github.com/GaspareG/ParallelPrefix).

5.1 Sequential algorithm

The sequential implementations

5.2 Block-based algorithm

TODO

5.3 Circuit-based algorithm

TODO

6 Experimental validation

6.1 experiments details

6.2 benchmark results

7 Conclusion